

SECOND ASSIGNMENT

Group number : *Acme-30*

Student Names and Numbers

Emanuele Santo Iaia 1924549

Simone Giordano 1772347



SAPIENZA
UNIVERSITÀ DI ROMA

Initial Brainstorming	1
Road-Warrior VPN	2
Wireguard Schema	2
VPN server	2
Wireguard Installation	2
Key generation	3
VPN server setup	3
Firewall rules	4
Start the VPN server	4
VPN client	4
Main-Internal VPN Tunnel	6
OPENVPN Schema	6
VPN Server	6
VPN Client	7
Test of the Configuration	8
Road-Warrior VPN	8
VPN Server	8
Final Remarks	10

Initial Brainstorming

In the beginning, we chose to use the OpenVPN program for both the VPN setup because we already used it before and we knew what we were working on. After the implementation of the VPN tunnel between the two routers of the network, with the openvpn GUI provided by opnSense, and after a chat with the professor, we opted to use the Wireguard program for the implementation of the Road-Warriors VPN, thanks to its “ready-out-of-the-box” approach and to use a new program to augment our VPN view.

Road-Warrior VPN

Following the previous paragraph, we have chosen to use Wireguard VPN to implement the Road-Warrior VPN in our network.

Wireguard Schema

The schema that we followed is the following:

On the Proxy server, that in this case is used as a VPN server with a new ip address 10.10.6.0/24

ip of eth0 interface	100.100.6.3
eth0 UDP port	51994
wg0 private IP	100.100.253.1
wg0 private key	2E7CqR1LKdt0U/ynNrPtAxCqTLNwsftNjyyn+EO/+0I=
wg0 public key	YhTMtByzkrJhjrBvbuTQAu/p7L2mUn6xF10DH7zwyW0=

On the Client server that we configured on the host pc, the ip of the interface is the one of the tap0, that is the address of the VPN that we connect to the acme network. For every user, the parts that are changing are the wg0 IP and the keys

ip of tap0 interface	100.101.0.3
wg0 private IP	100.100.253.2
wg0 private key	8IncOO3nxF3fDpFP6hgeTWxllJr/jl9H60gGpLVIWA=
wg0 public key	kTZBars9sE99QwrlHC8zYEAwD6tR/IO5fgzOZKY3sj0=

VPN server

Wireguard Installation

First of all we installed the Wireguard program on the Proxy server with the following commands:

- `echo "deb http://deb.debian.org/debian/ unstable main" | sudo tee /etc/apt/sources.list.d/unstable-wireguard.list`

- `printf 'Package: *\nPin: release a=unstable\nPin-Priority: 150\n' | sudo tee /etc/apt/preferences.d/limit-unstable`
- `sudo apt update`
- `sudo apt install wireguard`

Key generation

To generate the keys, we moved in the wireguard folder, `/etc/wireguard/`, and we executed this command:

```
umask 077; wg genkey | tee privatekey | wg pubkey > publickey
```

And the keys have been generated, which are the `publickey` and the `privatekey`.

VPN server setup

To correctly setup the VPN server, we need to create and later edit the `wg0.conf` file with the following directives that are consistent with the schema previously presented:

[Interface]

Address = `100.100.253.1/24`

SaveConfig = `true`

ListenPort = `51194`

PrivateKey = `2E7CqR1LKdt0U/ynNrPtAxCqTLNwsftNjyyn+EO/+0l=`

[Peer]

PublicKey = `kTZBars9sE99QwrIHC8zYEAwD6tR/IO5fgzOZKY3sj0=`

AllowedIPs = `100.100.253.2/32`

Endpoint = `100.101.0.3:45234`

[Peer]

PublicKey = `kTZBars9sE99QwrIHC8zYEAwD6tR/IO5fgzOZKY3sj0=`

AllowedIPs = `100.100.253.3/32`

[Peer]



















PublicKey = `kTZBars9sE99QwrIHC8zYEAwD6tR/IO5fgzOZKY3sj0=`

AllowedIPs = `100.100.253.4/32`

It could be possible to add more clients, hanging more [Peer] with the correspondent informations.

Firewall rules

About the road-warrior set-up, we enabled the VPN Traffic aimed to the Proxy server with the destination 100.100.6.3 and port 51194 in IN direction at the wan interface.

FIREWALL: RULES: WAN							Select category		
		Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description ⓘ
									Automatically generated rules
	  	IPv4 TCP	100.101.0.3/24	*	*	80 - 443	*	*	Connect from the host machine
	  	IPv6 TCP/UDP	*	*	2001:470:b5b8:1e00:40fa:57ff:fe4a:2073	80 - 443	*	*	Web Service (IPv6)
	  	IPv4 TCP/UDP	*	*	100.100.6.2	80 - 443	*	*	Web Service
	  	IPv4 UDP	*	*	100.100.6.3	51194	*	*	VPN road-warrior
	  	IPv4 *	*	*	*	*	*	*	Block Traffic
	  	IPv6 *	*	*	*	*	*	*	Block Traffic (IPv6)

Start the VPN server

To start the VPN server, on the proxy server, the following commands are executed:

- `sudo systemctl enable wg-quick@wg0`
- `sudo systemctl start wg-quick@wg0`

and to verify that everything is working correctly, we executed the `systemctl status` command and, with the `ip a` command, we verified that the interface has been correctly configured.

VPN client

Regarding the VPN client, the installation of the wireguard program and the key generation are the same executed on the server.

The main difference between the two is the `wg0.conf` file, in fact in the client those are the directives, that follows the schema:

Huck:

[Interface]

This Desktop/client's private key

PrivateKey = 8IncN003nxF3fDpFP6hgeTWxIIJr/jl9H60gGpLVIWA=

Client ip address

Address = 100.100.253.2/24

[Peer]

Debian 10 server public key

PublicKey = YhTMtByzkrJhjrBvbuTQAu/p7L2mUn6xF10DH7zwyW0=

set ACL

AllowedIPs = 100.100.253.0/24

Your Debian 10 LTS server's public IPv4/IPv6 address and port

Endpoint = 100.100.6.3:51194

Key connection alive

PersistentKeepalive = 20

Becca:

[Interface]

This Desktop/client's private key

PrivateKey = KJKFmm33J+F924liiZ9S8JPLj2hudtKxe/QkT1nMp1E=

Client ip address

Address = 100.100.253.3/24

[Peer]

Debian 10 server public key

PublicKey = YhTMtByzkrJhjrBvbuTQAu/p7L2mUn6xF10DH7zwyW0=

set ACL

AllowedIPs = 100.100.253.0/24

Your Debian 10 LTS server's public IPv4/IPv6 address and port

Endpoint = 100.100.6.3:51194

Key connection alive

PersistentKeepalive = 20

Gim

[Interface]

This Desktop/client's private key

PrivateKey = KLqmXlzfQzs59j1s59E618bt5Y87PqO+iq817HA1pF4=

Client ip address

Address = 100.100.253.4/24

[Peer]

Debian 10 server public key

PublicKey = YhTMtByzkrJhjrBvbuTQAu/p7L2mUn6xF10DH7zwyW0=

set ACL

```
AllowedIPs = 100.100.253.0/24
## Your Debian 10 LTS server's public IPv4/IPv6 address and port ##
Endpoint = 100.100.6.3:51194
## Key connection alive ##
PersistentKeepalive = 20
```

To start the VPN client, the same command for the server have been used.

Main-Internal VPN Tunnel

For the solution of the VPN Tunnel between the Main router and the internal one, we opted for the implementation of OpenVPN through the GUI provided by the OPNSense interface of the routers.

OPENVPN Schema

Regarding the Main firewall

IP of the interface heading the Internal	100.100.254.1
IP of Tunnel network	10.10.94.1

Regarding the Internal firewall

IP of the interface heading the Internal	100.100.254.2
IP of Tunnel network	10.10.94.2

VPN Server

In the configuration page of the VPN Server we inserted the following characteristics:

General Information

- Server mode: Peer to Peer via Shared key
- Protocol: UDP
- Device mode: tun
- Interface: Internal

- Local Port: 1194

Cryptographic Settings

- Generation of a Shared key
- Encryption Algorithm: AES-256-CBC
- Auth Digest Algorithm: SHA1

Tunnel Settings

- IPv4 Tunnel network: 10.10.94.0/24
- IPv4 Local network (that are the networks that will be accessible from the internal router): 100.100.4.0/24, 100.100.6.0/24
- IPv4 Remote network (that are the networks that will be accessible to the Main router): 100.100.1.0/24, 100.100.2.0/24

After this configuration the VPN server has been started.

VPN Client

For the configuration of the VPN Client, that in our case is the internal router, the shared key generated by the main router in precedence has been copied.

The configuration is similar to the one seen before, with the main difference being that in the client, a remote server has been specified, that is the main router 100.100.254.1:1194 and the shared key is not generated by it, but it is the same copied before.

To prove that the Client has been correctly configured, in OPNSens there is a useful tool that visualize the connection status of the OpenVPN clients:

VPN: OpenVPN: Connection Status						
Client Instance Statistics						
Name	Remote Host	Virtual Addr	Connected Since	Bytes Sent	Bytes Received	Status
OpenVPN client UDP	100.100.254.1	10.10.94.2	2021-04-30 08:57:56	2.92 MB	37.52 MB	up

Moreover, both the VPN client and server, based on the the Remote networks added in the configuration, have added a new route to the router that forces to use the VPN tunnel created if it wants to reach them:


```
openvpn[33452] /sbin/route add -net 100.100.2.0 10.10.94.2 255.255.255.0
```

```
openvpn[33452] /sbin/route add -net 100.100.1.0 10.10.94.2 255.255.255.0
```

Test of the Configuration

Road-Warrior VPN

As regarding the test configuration of the Road-Warrior VPN we choose to test it using the packet capture tool, indeed we send ping from one of the clients to the VPN server capturing all the echo request coming from the WAN interface of the Main Firewall. Analyzing those packet with Wireshark we noticed that the packet's payload was encrypted by Wireguard.

Apply a display filter ...<?/?>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	100.101.0.3	100.100.6.3	WireG...	170	Transport Data, receiver=0x82395920, counter=3, datalen=96
2	0.000679	100.100.6.3	100.101.0.3	WireG...	170	Transport Data, receiver=0x9E8A8BEC, counter=2, datalen=96
3	1.003264	100.101.0.3	100.100.6.3	WireG...	170	Transport Data, receiver=0x82395920, counter=4, datalen=96
4	1.003895	100.100.6.3	100.101.0.3	WireG...	170	Transport Data, receiver=0x9E8A8BEC, counter=3, datalen=96
5	2.006616	100.101.0.3	100.100.6.3	WireG...	170	Transport Data, receiver=0x82395920, counter=5, datalen=96
6	2.007089	100.100.6.3	100.101.0.3	WireG...	170	Transport Data, receiver=0x9E8A8BEC, counter=4, datalen=96
7	3.010807	100.101.0.3	100.100.6.3	WireG...	170	Transport Data, receiver=0x82395920, counter=6, datalen=96
8	3.011359	100.100.6.3	100.101.0.3	WireG...	170	Transport Data, receiver=0x9E8A8BEC, counter=5, datalen=96
9	4.011304	100.101.0.3	100.100.6.3	WireG...	170	Transport Data, receiver=0x82395920, counter=7, datalen=96
10	4.011996	100.100.6.3	100.101.0.3	WireG...	170	Transport Data, receiver=0x9E8A8BEC, counter=6, datalen=96

> Frame 1: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits)
> Ethernet II, Src: ee:be:55:e2:7c:15 (ee:be:55:e2:7c:15), Dst: 76:61:8c:b4:d1:12 (76:61:8c:b4:d1:12)
> Internet Protocol Version 4, Src: 100.101.0.3, Dst: 100.100.6.3
> User Datagram Protocol, Src Port: 36032, Dst Port: 51194
> WireGuard Protocol
Type: Transport Data (4)
Reserved: 000000
Receiver: 0x82395920
Counter: 3
Encrypted Packet

```

peer: YhTMtByzkrJhjrBvbuTQAu/p7L2mUn6xF10DH7zwyW0=
endpoint: 100.100.6.3:51194
allowed ips: 100.100.253.0/24
latest handshake: 1 minute, 36 seconds ago
transfer: 61.88 KiB received, 374.47 KiB sent
persistent keepalive: every 20 seconds
root@katha:/etc/ssh# ping 100.100.253.1
PING 100.100.253.1 (100.100.253.1) 56(84) bytes of data.
64 bytes from 100.100.253.1: icmp_seq=1 ttl=64 time=10.2 ms
64 bytes from 100.100.253.1: icmp_seq=2 ttl=64 time=11.9 ms
64 bytes from 100.100.253.1: icmp_seq=3 ttl=64 time=10.0 ms
^C
--- 100.100.253.1 ping statistics ---

```

VPN Server

As regarding the test configuration of VPN server we choose to test it using the packet capture and traceroute tools, indeed we first send pings from internal host to external hosts listening on the internal interface of the Main Firewall capturing all the echo requests coming from the internal network, analyzing those packets with wireshark we noticed that the packet's payload was encrypted by OpenVPN.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	100.100.254.1	100.100.254.2	OpenV...	110	MessageType: Unknown Messagetype[Malformed Packet]
2	3.302161	100.100.254.1	100.100.254.2	OpenV...	686	MessageType: Unknown Messagetype[Malformed Packet]
3	3.317303	100.100.254.1	100.100.254.2	OpenV...	270	MessageType: Unknown Messagetype[Malformed Packet]
4	3.384040	100.100.254.2	100.100.254.1	OpenV...	174	MessageType: P_DATA_V2
5	3.384365	100.100.254.1	100.100.254.2	OpenV...	174	MessageType: Unknown Messagetype[Malformed Packet]
6	4.398892	100.100.254.2	100.100.254.1	OpenV...	174	MessageType: Unknown Messagetype
7	4.399184	100.100.254.1	100.100.254.2	OpenV...	174	MessageType: Unknown Messagetype[Malformed Packet]
8	5.470257	100.100.254.2	100.100.254.1	OpenV...	174	MessageType: Unknown Messagetype[Malformed Packet]
9	5.470504	100.100.254.1	100.100.254.2	OpenV...	174	MessageType: Unknown Messagetype[Malformed Packet]
10	5.507006	100.100.254.1	100.100.254.2	OpenV...	142	MessageType: Unknown Messagetype[Malformed Packet]
11	5.521969	100.100.254.1	100.100.254.2	OpenV...	142	MessageType: Unknown Messagetype[Malformed Packet]
12	5.523368	100.100.254.1	100.100.254.2	OpenV...	142	MessageType: Unknown Messagetype[Malformed Packet]
13	5.537016	100.100.254.1	100.100.254.2	OpenV...	142	MessageType: Unknown Messagetype[Malformed Packet]
14	5.538194	100.100.254.1	100.100.254.2	OpenV...	142	MessageType: Unknown Messagetype[Malformed Packet]
15	5.539005	100.100.254.1	100.100.254.2	OpenV...	142	MessageType: Unknown Messagetype[Malformed Packet]
16	5.539707	100.100.254.1	100.100.254.2	OpenV...	142	MessageType: Unknown Messagetype[Malformed Packet]
17	5.552439	100.100.254.1	100.100.254.2	OpenV...	142	MessageTvpne: Unknown Messagetype[Malformed Packet]

> Frame 1: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
 > Ethernet II, Src: d2:cb:c3:2b:36:7e (d2:cb:c3:2b:36:7e), Dst: 3a:98:dd:40:99:44 (3a:98:dd:40:99:44)
 > Internet Protocol Version 4, Src: 100.100.254.1, Dst: 100.100.254.2
 > User Datagram Protocol, Src Port: 1194, Dst Port: 56803
 > OpenVPN Protocol
 > [Malformed Packet: OpenVPN]
 > [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
 > [Malformed Packet (Exception occurred)]
 > [Severity level: Error]
 > [Group: Malformed]

Then we used Traceroute tool to check the route of the packets sent by internal network to external network and we verified that the hop used to pass between the networks are the tunnel's interface (Main Firewall side > 10.10.94.1 | Internal Firewall side > 10.10.94.2)

```
# /usr/sbin/traceroute -w 2 -n -m '3' -s '100.100.2.1' '100.100.6.2'
traceroute to 100.100.6.2 (100.100.6.2) from 100.100.2.1, 3 hops max, 40 byte packets
 1  10.10.94.1  0.941 ms  0.724 ms  0.632 ms
 2  * * *
 3  * * *
```

Final Remarks

This is our proposed solution useful to configure the Road-Warrior VPN and the Main-internal VPN. Those topics were very interesting as the two topics covered are very frequent in the real world as matter of fact we decided to configure the vpn in our house thanks to the “OpenVPN cloud” services in order to navigate on the web through a secure VPN connection and connect to our home pc when we are abroad with “OpenVPN connector”.